Box 2D
JOINTS

INDEX

- **Box2d & Box2D Joints Definition**
- **Basic Concepts And Definitions**
- **Box2D Joint types**
- **Classes that I need**
- **Example**

BOX2D & BOX2D JOINTS DEFINITION

BOX2D:

- Is A Physics Library For 2D Games. It Is Written In C++ (We Use A Wrapper Like Java)

BOX2D JOINTS:

- A Physics Concepts To Constrain Bodies To The World Or To Each Other. Joints Can Be Combined In Many Different Ways To Create Motions.
- Each Joint Type Has A Definition That Derives From **B2JointDef**. All Joints Are Connected Between Two Different Bodies. One Body May Static.
- You Can Specify User Data For Any Joint Type And You Can Provide A Flag To Prevent The Attached Bodies From Colliding With Each Other. This Is Actually The Default Behavior And You Must Set **The CollideConnected Boolean** To Allow Collision Between To Connected Bodies.

Basic Concepts
&
Definitions

PHYSICS WITH BOX2D

Basic Concepts And Definitions - I

World

A physics world is a collection of bodies, fixtures, and constraints that interact together.

Body (Rigid Body)

A chunk of matter that is so strong that the distance between any two bits of matter on the chunk is constant. They are hard like a diamond.

Shape

A shape is 2D geometrical object, such as a circle or polygon.

PHYSICS WITH BOX2D

Basic Concepts And Definitions - II

Fixture

A fixture binds a shape to a body and adds material properties such as density, friction, and restitution. It also puts a shape into the collision system.

Forces and Impulses

You can apply forces, torques, and impulses to a body. When you apply a force or an impulse, you provide a world point where the load is applied. This often results in a torque about the centre of mass.

Joint

This is a constraint used to hold two or more bodies together.

Some times it is necessary to define a joint limit and a joint motor.

PHYSICS WITH BOX2D

Body Types

Static

A static body does not move under simulation and behaves as if it has infinite mass.

Kinematic

A kinematic body moves under simulation according to its velocity. They do not respond to forces and do not collide with other kinematic or static bodies.

Dynamic

A dynamic body is fully simulated. They can be moved manually by the user, but normally they move according to forces. It can collide with all body types and it has finite, non-zero mass.

Box2D Joint TYPES

JOINT DESCRIPTIONS

Types:

Distance

Makes sure that the distance between two points on two bodies remain constant. Can be made “soft” to observe a springboard-like behavior.

Revolute

Acts like a hinge, such as how a door is connected to a wall. Exemplified in the code example.

Weld

Links two bodies together to act as one body. Provides a constraint for all relative motion between two bodies. Example would be like two lego bricks being hooked together

JOINT DESCRIPTIONS

Types:

Prismatic

It allows for a body to move/slide along a defined axis. This can be useful for creating elevators.

Pulley

The pulley connects two bodies to the ground and to each other. As one body moves up, the other body will receive the slack and move down.

Rope

Restricts the maximum distance between two bodies, even under high-load. This allows for proper simulation of ropes, so that they may not stretch beyond their limit.

Gear

The gear joint can only connect revolute and/or prismatic joints. Better than making your own gear polygon shapes, as it offers better flexibility along with efficiency.

Wheel

The wheel joint restricts a point on bodyB to a line on bodyA. The wheel joint also provides a suspension spring for simulating real wheel suspension with little coding work.

* Open and Watch video **“VBox2D_preview_joints_Box2D_editor”**

Classes That I need

PHYSICS WITH BOX2D

Steps To Use Box2D - I

- Define the World.

```
private World world;
…
@Override
public void create() {
    final float GRAVITY_ACCELERATION = -9.5f;
    world = new World(new Vector2(0, GRAVITY_ACCELERATION), true);
    …
}
```

- Define a Box2DDebugRenderer and OrthographicCamera to draw a schema of the world.

```
private Box2DDebugRenderer b2DRenderer;
private OrthographicCamera camera;
…
@Override
public void create() {
    …
    b2DRenderer = new Box2DDebugRenderer();
    camera = new OrthographicCamera(width/2, Height/2);
    …
}
```

PHYSICS WITH BOX2D

Steps To Use Box2D - III

- Define bodies.

```
public static float PPM = 32; //PIXELS PER METER

public Body createBox(int x, int y, int width, int height, boolean isStatic, boolean
fixedRotation, float density) {

    Body pBody;
    BodyDef def = new BodyDef();

    if (isStatic) {
        def.type = BodyDef.BodyType.StaticBody;
    } else {
        def.type = BodyDef.BodyType.DynamicBody;
    }

    def.position.set(x / PPM, y / PPM);
    def.fixedRotation = fixedRotation;
    pBody = world.createBody(def);

    PolygonShape shape = new PolygonShape();
    shape.setAsBox(width / 2 / PPM, height / 2 / PPM);

    pBody.createFixture(shape, density);
    shape.dispose();
    return pBody;
}
```

- Define **B2JointDef**.(Rope)

```
public void buildRopeJoints() {
    ArrayList<Body> bodies = new ArrayList<>();
    int inicio = 56;
    bodies.add(createBox(0, inicio, 30, 32, true, true, 1));

    for (int i = 1; i < 5; i++) {
        bodies.add(createBox(0, (-i * 32) + inicio, 4, 10, false, false, 1));

        RopeJointDef rDef = new RopeJointDef();
        rDef.bodyA = bodies.get(i - 1);
        rDef.bodyB = bodies.get(i);
        rDef.collideConnected = true;
        rDef.maxLength = 0.5f;

        rDef.localAnchorA.set(0, -0.25f);
        rDef.localAnchorB.set(0, 0.25f);

        world.createJoint(rDef);
    }
}
```

- Define **B2JointDef**.(Rope)

```
public void createBridge() {
    ArrayList<Body> bodies = new ArrayList<>();

    bodies.add(createBox(400, -180, 32, 32, true, true, 1));

    for (int i = 1; i <= 20; i++) {

        if (i == 20) {
            bodies.add(createBox(850, -180, 32, 32, true, true, 1));
        } else {
            bodies.add(createBox(19 * 32, 0, 14, 10, false, false, 0f));
        }

        RopeJointDef rDef = new RopeJointDef();
        rDef.bodyA = bodies.get(i - 1);
        rDef.bodyB = bodies.get(i);
        rDef.collideConnected = true;
        rDef.maxLength = 0.8f;

        rDef.localAnchorA.set(0, -0.25f);
        rDef.localAnchorB.set(0, 0.25f);

        world.createJoint(rDef);
    }

}
```

- Define **B2JointDef**.(Pulley)

```
public void buildPulleyJoints() {
    Body bodyA = createBox(-100, -30, 30, 32, false, true, 1);
    Body bodyB = createBox(100, 30, 30, 32, false, true, 1);

    bodyA.setLinearDamping(1.75f);
    bodyB.setLinearDamping(1.75f);

    PulleyJointDef pDef = new PulleyJointDef();
    pDef.bodyA = bodyA;
    pDef.bodyB = bodyB;
    pDef.collideConnected = false;
    pDef.localAnchorA.set(0, 0);
    pDef.localAnchorB.set(0, 0);
    pDef.lengthA = 4;
    pDef.lengthB = 2;
    pDef.groundAnchorA.set(-3, 4);
    pDef.ratio = 1f;

    pDef.groundAnchorB.set(2, 4);

    world.createJoint(pDef);
}
```

PHYSICS WITH BOX2D

Steps To Use Box2D - VI

- In the render method, we have to simulate the physics equations at discrete points of time calling step method. It should be called each 1/60 sg.

```
@Override
public void render(float delta) {
  Gdx.gl.glClearColor(0f,0f,0f,1f);
  Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

  final int VELOCITY_ITERATIONS = 6;
  final int POSITION_ITERATIONS = 2;

  world.step(delta, VELOCITY_ITERATIONS, POSITION_ITERATIONS);

  b2DRenderer.render(world, camera.combined);
}
```

EXAMPLE
(Android Studio)

Any Question?